OCaml Companion Tools

Xavier Clerc – forum@x9c.fr

July 25, 2012

Abstract The objective of this talk is to present several tools that aim to ease the development of software written with the OCaml language. These tools are particularly suited to help the developer during the debugging phase. Although this part of the development process is often overlooked in the functional community, such tools can dramatically reduce development time.

Process

We propose a debugging process based on the following four steps, each step being backed up by a distinct tool:

1. actually identifying the bugs, with the Kaputt testing library;
2. understanding what is going on, with the Bolt logging library;
3. asserting what is effectively tested, with the Bisect coverage tool;
4. checking that the revised source is homogeneous and contains no *code smell*, with the Mascot program.

The Kaputt testing library

Kaputt features four kinds of tests:

- assertion-based tests, inspired by the xUnit tools;
- specification-based tests, inspired by the *QuickCheck* tool;
- enumeration-based tests, inspired by the *SmallCheck* tool;
- shell-based tests.

When writing assertion-based tests, the developer encodes input values and checks that output values satisfy given assertions. When writing specification- or enumeration-based tests, the developer encodes the specification of the tested function and then requests the library to generate

random values to be tested against the specification. Finally, shell-based tests are barely a replacement for shell scripts executing programs and comparing their outputs to a previously checked reference.

The Bolt logging library

Bolt is inspired by and modeled after the famous log4j logging framework for Java. Bolt is highly customizable and features a camlp4-based syntax extension (introducing a "`LOG msg LEVEL lvl`" expression) that allows to remove all logging statements to produce executables that do not incur any performance penalty for logging statements used during the development stage.

The Bisect coverage tool

Bisect is implemented as a camlp4-based tool that allows to instrument your application before running tests. After application execution, it is possible to generate a report in HTML format that is the replica of the application source code annotated with code coverage information.

The Mascot style-checker tool

Mascot provides checks in various categories such as code, documentation, interface, metrics, and typography. The goal of the tool is to allow a (team of) developer(s) to enforce style properties over a source codebase for greater coherency and style uniformity. The tool is highly customizable, allowing one to choose the checks to perform, as well as exceptions to those checks in given source files. Finally, a plugin system allows the developer to provide custom checks.

Bonus track: the search feature of the Argot tool

During development, it is not unusual to look for a symbol either by its name, or by its type. Argot allows to search for elements by exact name, by regular expression over names, or by type. When using type-based search, the lookup is actually done up to type isomorphism. This theoretical statement has a precise practical implication: the developer does not have to know the order of parameters to a function, whether the function is currified or tuplified, *etc.*